Indian Journal of Science and Technology, *Vol 12(16), DOI: 10.17485/ijst/2019/v12i16/143833, April 2019*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Proposed Low Cost Library Automation in Developing Countries Using Modern Programming and Information Technologies

Naveed-e-Sehar[1*], Humera Tariq[2], Sayaam Qazi[2] and Aqil Burney[3]

[1]Department of Library and Information Sciences, University of Karachi, Karachi, Karachi City, Sindh 75270, Pakistan; nsehar@uok.edu.pk
[2]Department of Computer Science, University of Karachi, Karachi, Karachi City, Sindh 75270, Pakistan; humera@uok.edu.pk
[3]College of Computer Science and Information Systems (CCSIS), Institute of Business Management, Korangi Creek, Karachi, Karachi City, Sindh 75190, Pakistan

Abstract

Background/Objectives: Manual Record Keeping results in lot of inconvenience for readers and librarians. The objective of this study is to explore the automation possibility of public libraries in developing countries. **Methods/Statistical Analysis:** Two-part solution using Modern Programming Platforms has been proposed to the above stated problem: 1. A cross platform application with a local DataStore and Sync capabilities is needed which will be installed on every target library and operated by a librarian, and 2. A server is required to store a Global Datastore derived from all the active local datastores connected to the server, from which users can query information through webpage. **Findings:** Proposed automation not only supports all Operating Systems namely MS-Windows, Linux and MacOS but it also supports existing APIs on Java and C++. For JAVA, separate VM needs to install but it is fully supported on C# due to usage of Electron Framework. **Improvements/Applications:** All the interactions between reader and librarian are automated successfully and authorized user time is saved due to supported browsing at home. The automation helps in trending analysis about particular book, article and periodical reading.

Keywords: Csharp, C++, Electron Frame Work, Library Automation, Software, JAVA

1. Introduction

Since the advent of Information technology many of repetitive and cumbersome jobs are being done by automated systems because computers are very good at doing same thing over and over again with unmatched efficiency while not making mistakes. Unfortunately, there are still many disciplines in developing countries, which haven't gotten their share in this era of technology, one of which is libraries. There are still many libraries where librarians have to keep records in registers by hand. Human error creates even more problems because it is hard to undo a mistake on pen and paper based record. Also there is no way for a book reader to know if a certain book is available in a particular library, or even if it exists; Is it already borrowed by someone else? There are some systems that have tried to address these problems but did not succeed in achieving their ultimate goal. In this paper we discuss the limitations of such systems that already exist and not only propose infact implement a system that improves upon the weaknesses of said systems. Since the data collected through our system will be in managed form we also touch on the subject of utilizing that data to improve and predict behavioral patterns of the readers. There have been some efforts to automate the library management but they fall short on many grounds which are essential to such systems. Any computer based automation system should at least have following traits to make it of any value. 1. User experience, 2. Robustness, 3. Platform agnostic, and 4. Connectivity. User experience is undoubtedly the most essential part of computer software. If the user does not

feel comfortable using the software, then it is of no use at all. Many currently existing softwares fall short on this part[1]. A management system should not just work in normal scenarios. It should be robust enough to handle at least some of the unforeseen scenarios. Fortunately, this part is usually covered by current systems by help of features like backup and restore[2]. Since the world of software is so diverse with respect to both software and hardware good software should work on as many varieties as possible to appeal to a large size of target audience. This is the weak spot of current library automation systems; they work on very selective system causing frustration to the users[4,6,15]. In today's world of internet our lives revolve around the availability of information on the go. For example, a reader should be able to know from his reading desk if a book he intends to borrow from library even exists there or not. Unfortunately, there is no such system in existence. In current state a reader must go to library to find out if the book is available or not for him to borrow. Our solution will address that problem too[7,8,14].

2. Existing Solution

Koha[9,10] is a well known tool for library management. In this section we look at its weaknesses and discuss why it is still not a complete solution to library management problem. In our thorough study of the tool we found that it suffers the most in following aspects of an ideal library management system. 1. User experience: When paired with mediocre user experience even the most functional software can become a chore to use. Case in point Koha suffers from this exact problem. Although it has probably the richest feature set among the alternatives, it does not have a UI whose design is based on modern UX principles, 2. Platform agnostic: We have already discussed the importance of a tool being platform agnostic so its users are not stuck into an opinionated ecosystem. Koha does not available on every major platform in the PC world. Above all that the platform that it supports (linux) is one of the least used operating system among the common computer users, and (3) Connectivity: Every passing day we see more and more apps move to the internet since users want to be able to access their data from wherever they want. Koha suffers on this front too. It is just a local storage system for books in this aspect. There is no synchronization of data in the server so users can browse through it effortlessly on the go.

3. Methodology

In order to deeply understand the problem, we conducted many surveys to collect relevant data[3]. After tireless analysis of the collected data we have dug deep into the core problem which is two-part statement described as follows: 1. Librarians need a system to store, retrieve, edit and delete books data along with the information about the readers who borrowed them, and 2. Readers should have a system from which they can ask about availability and other relevant information about books without having to visit the library. The proposed system will have four components listed here and are shown in Figure 1. The integral components of the system are: Local database, Global database, Create Record Update Delete (CRUD) application and Users front-end. We will discuss the detailed design of all these components in order. For each component we will discuss: 1. Role in the complete system - What is the core purpose of the component and how it works as a part of the whole system, 2. Specifications - Specification are the attributes of a components that describe upper and lower bounds of its capabilities. We will report all specifications in Result and Discussion Section, 3. Case Specific Design Choices - Since a lot of different results can be accomplished by applying different methods. It is crucial to choose the right one that fits the needs of the user of the system. This part will describe why things are the way they are backed by solid arguments, 4. Interactions with Other components - A component is nothing without the system in which it resides. It is important to know how it interacts with other components of the system. This part will describe exactly that, and 5. Functional requirements - This part explains the physical requirements for the component to work as expected.

3.1 Local Database

The system we are going to implement features a local database per library. For a database there are a plethora of options available to choose from each having their own set of advantages and disadvantages as presented in Table 1. Since proposed application does not need very complex API support we can drop the SQL and MongoDB options. This greatly simplifies the installation process for the end user. They don't have to worry about starting services in correct order to make system work properly. Now we were left with two options sqlite and neDB[12,13,16,17]. After extensive testing we went for neDB due to its faster

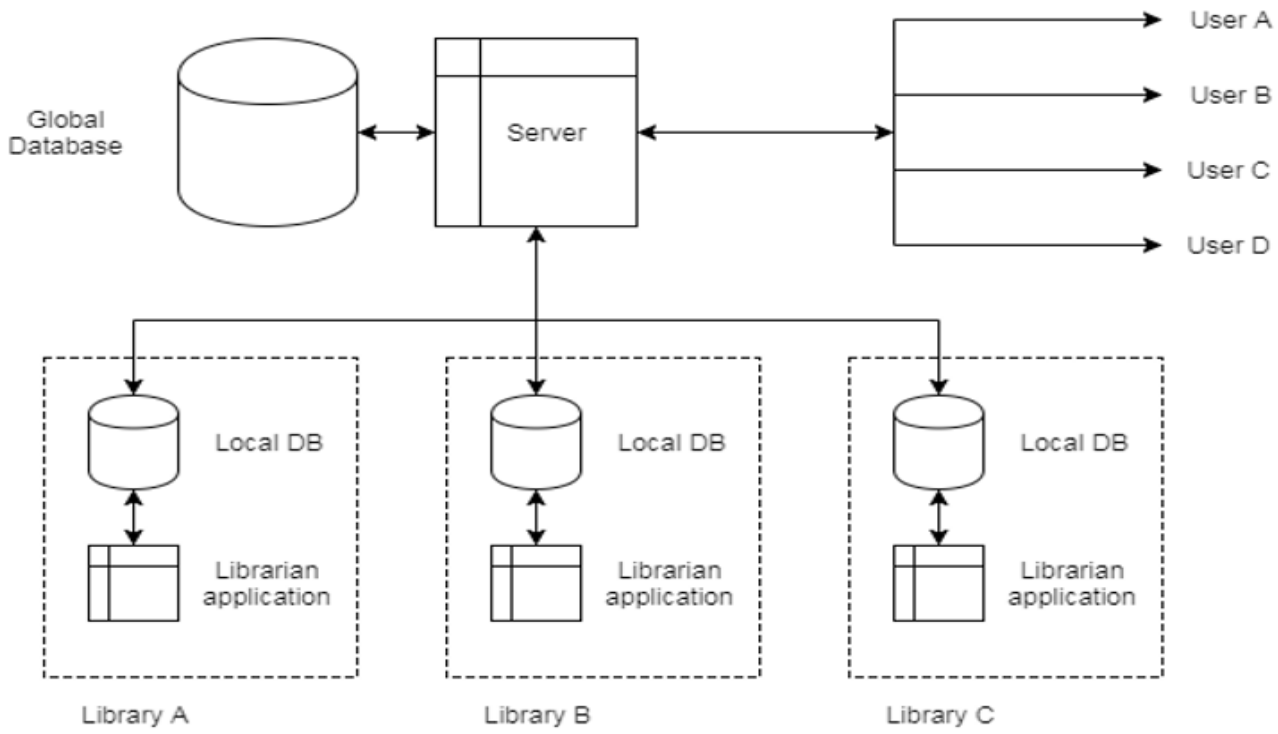

Figure 1. Components of proposed automated library system.

query response and seamless integration with the system. Application actually directly interacts with local database as shown in Figure 2 and 3. It can perform following operations on local database.

1. ADD a new book to records when they got new books.
2. UPDATE relevant data about a book (like it is being borrowed or returned).
3. DELETE a book that has been stolen or lost.
4. FIND a particular record in the database.

3.2 Global Database

Since every library has its own local database, there must be a way to combine all that data into a single searchable repository, on which users can perform queries. The functional requirements of this database are different from local database. In this case we had a lot of options to choose from. Table 2 describes some options. It is quite clear from Table 2 that we went with the NoSQL option. As the name suggests global database actually combines every local database of each library to form a universal source of information about the books present in the

Table 1. Choices for local database

Database	Advantages	Disadvantages
SQL	Extremely flexible and can handle huge quantities of data.	Very large package. Requires standalone installation process. Complex API
MongoDB	Simple API, efficient for flat data structures. Very fast	Large package. This also requires standalone installation.
sqlite	Moderately complex API, lightweight than its superset SQL	Known issues with some technologies. Medium footprint.
neDB	Simple API. Extremely efficient and very lightweight. Very small footprint.	Can only handle moderately sized databases.

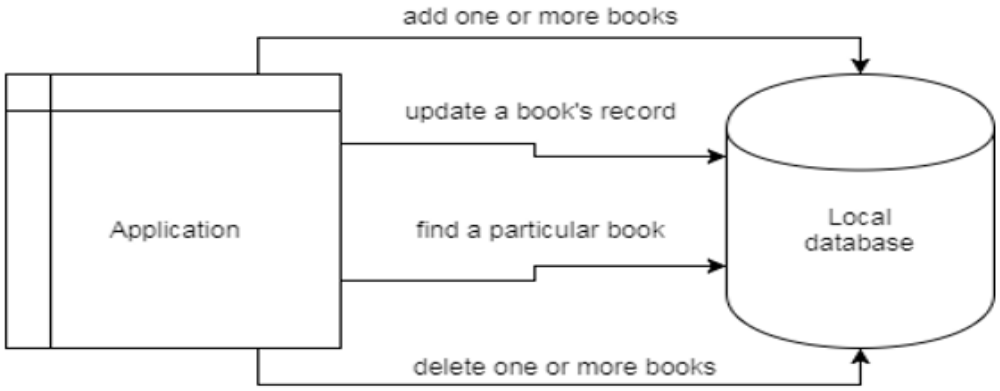

Figure 2. Interaction modeling with local database.

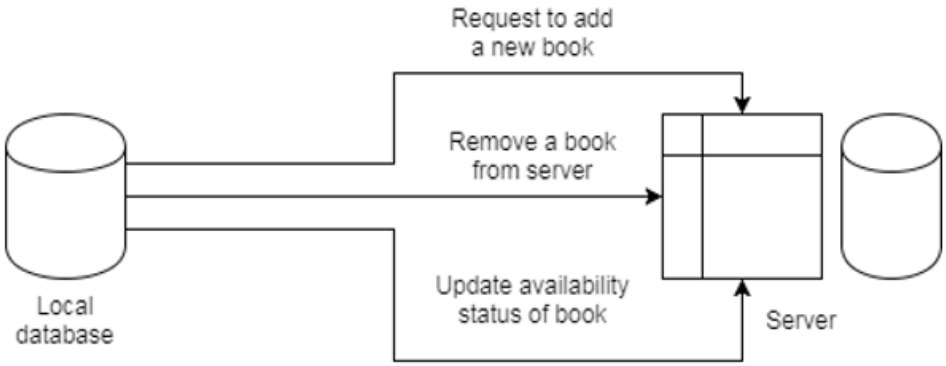

Figure 3. Interaction modeling with server.

whole area which can be a city, state, country or even the whole world. Global database mainly interacts with two parts of the system. 1. Front end webpage, and 2. Local Database.

3.2.1 Front End Webpage

Users visiting the front end of the global database server can perform following types of queries to the server and the server will provide the appropriate response. Figure 4 depicts and captured the described scenarios.

1. SEARCH a book i.e. does certain book exists in any library?
2. WHERE i.e. if the book exists, in which library it resides?
3. QUERY STATUS of a book i.e. if the book exists then is it available to be borrowed by the user or someone already borrowed it?

3.2.2 Local Database

The global database asks all the local databases at regular intervals, if the contents of it have changes and need to be

Table 2. Choices for global database

Database	Advantages	Disadvantages
SQL	Very flexible, support for very large databases	Complex querying mechanism causes a bit of overhead
NoSQL	Really simple yet flexible ORM based querying system. Easy to swap storage architecture.	Less mature than SQL but stable enough for our purposes

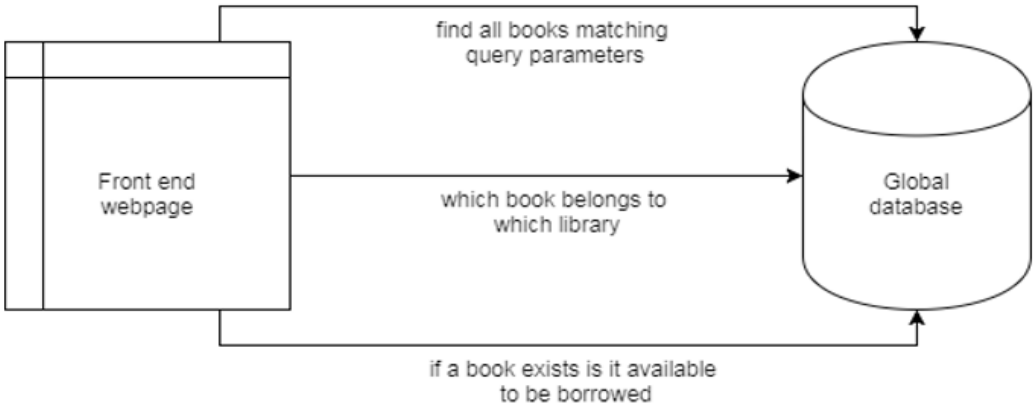

Figure 4. Interaction modeling of global DB with front end web page.

updated so that the data on global and local level remains consistent. Figure 5 shows these activities.

3.3 CRUD Application

CRUD application is arguably the most important part of this system. Since this part of the system where the librarian interacts it needed to be the most flexible of all. This is the part where other systems lacked the most. Three major Operating systems installed on almost every computer in the world namely: Microsoft windows, Linux or MacOS. We wanted to support all of these platforms to make it easy for the user to adapt our system without the hassle of installing new operating systems. We considered a lot of frameworks on which we could have built the CRUD. Table 3 shows the OS compatibility of different frameworks. Table 3, it is obvious that we had to go with the electron framework. It is cross platform that means once an app is written with electron it will work on every major platform. The user of this component is librarians. They may interact with the system in many different ways which are depicted in the Figure 6.

3.4 Front End Web Page

Since this part of the system is in the form of a webpage we had to find out what are the trends of browser usage nowadays which was not very hard because there are already people dedicated to this task. According to W3 Schools as of 2017[5], the usage statistics of web browsers turns out to be as given in Table 4. It is clear from Table 4 that if our web page works on Chrome and Firefox, we would have covered almost 90% of the world's internet users. That doesn't mean we leave other 10% out. We just prioritize the compatibility efforts according to user-base and we are happy to report that the final version of our web page works on 99.95% of the browsers, 0.05% being those who have JavaScript disabled on their browser. The user of this component is the readers who want to find or borrow books. Their interactions with the system are depicted in Figure 7.

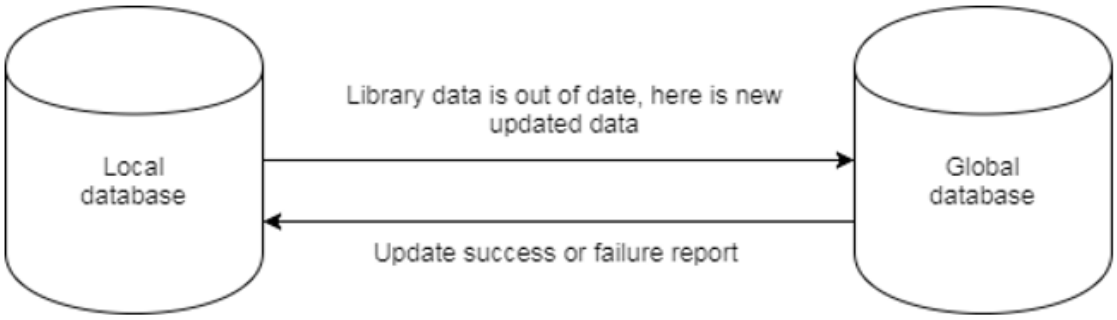

Figure 5. Sync of global DB with local DB.

Table 3. Choice for development with respect to OS

	Ms Windows	**Linux**	**MacOS**
C/C++	Different system APIs	Different system APIs	Different system APIs
Java	Same API but requires users to install VM separately	Same API but requires users to install VM separately	Same API but requires users to install VM separately
C#	Fully supported	Through hacky methods	Not supported
Electron	Full support	Full support	Full support

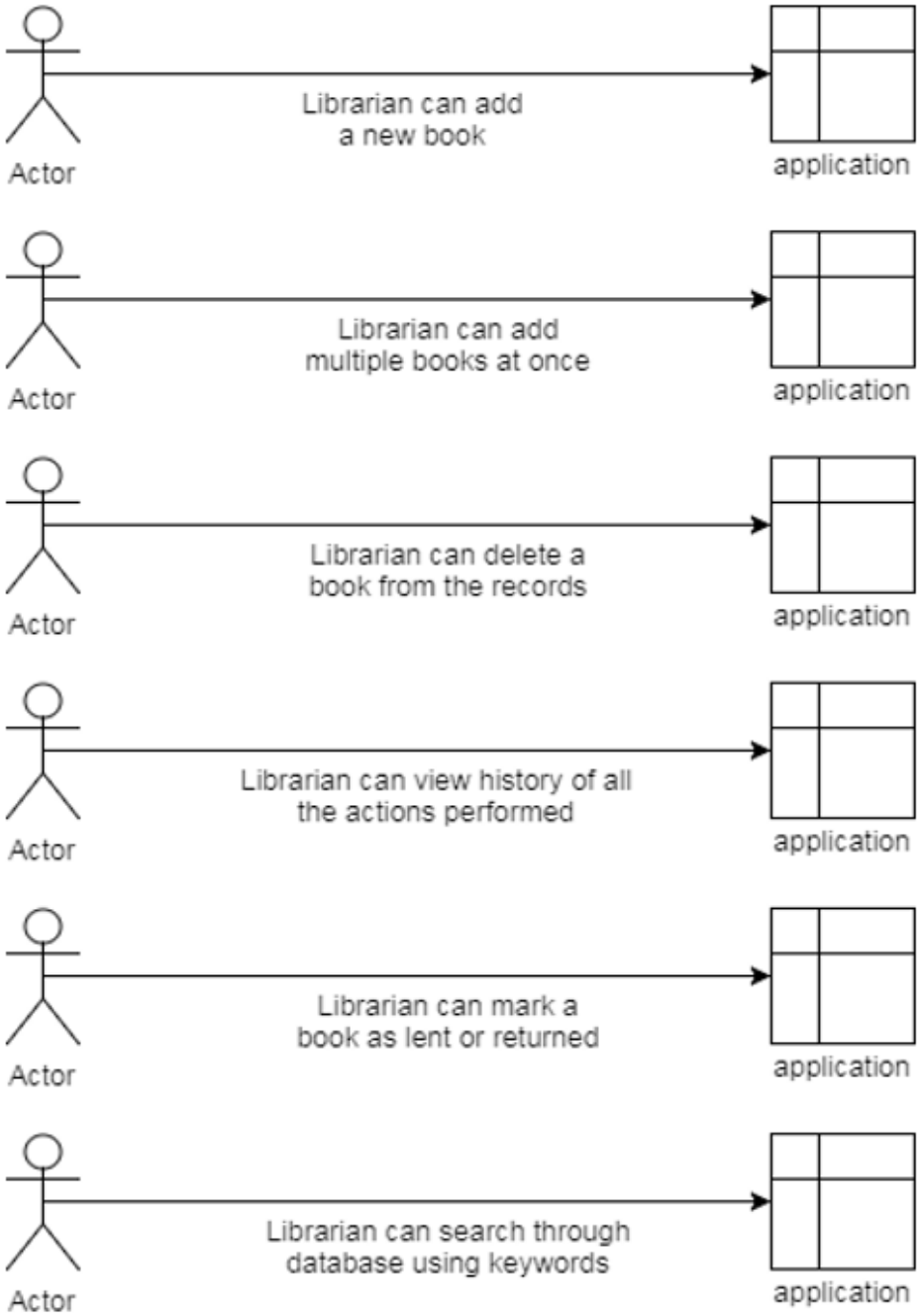

Figure 6. Librarian interaction modeling with CRUD application.

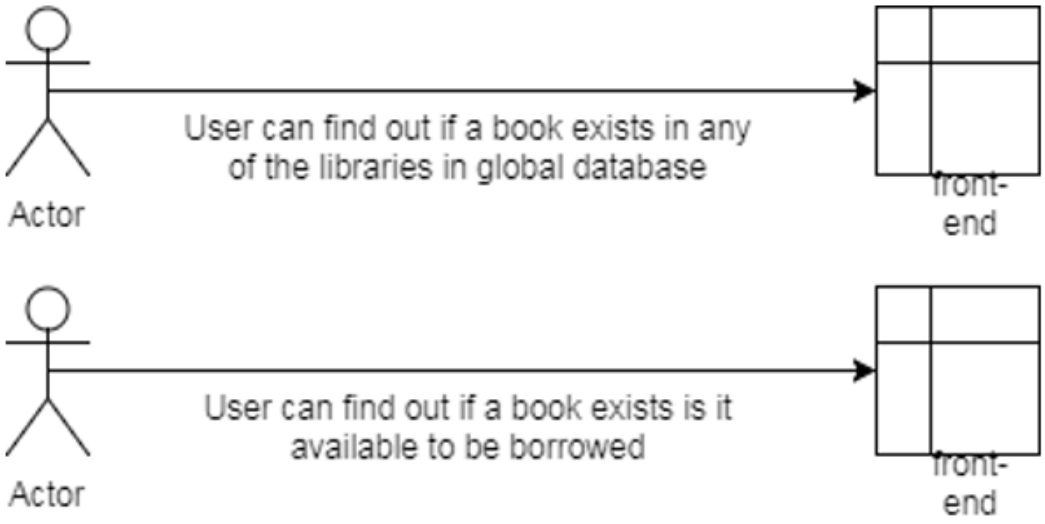

Figure 7. User interaction at front end.

Table 4. Choice for front end web page design

Browser	Percent user-base share
Chrome	76.9%
Internet explorer	4.3%
Safari	3.0%
Firefox	13.1%
Opera	1.2%

3.5 Results and Discussion

After thorough testing of our system we have concluded following performance benchmarks about the local database component. All benchmarks were taken on a computer with very common configuration. To Insert a record: it can perform 10,680 ops / sec; to find a record: it can perform 43,290 ops / sec; to update a record: it can do 8000 ops / sec; for deletion, it can perform 11,750 ops / sec.

4. Conclusion

We explore the possibility of connecting and automation of all public libraries in a developing country and proposed a low cost, simplified and an immense automation procedure. To fulfil those automation needs we designed and developed a multi-tier, responsive and cross platform client server application based on recent technologies. The web based system is implemented using typescript language along with NoSQL database at local systems.

5. Acknowledgement

We are thankful to Gulshan-e-Iqbal Library Administration especially Ms. Talat Shakeel for cooperation and software deployment. We are also thankful to Molvi Abdul Haque Library Aministration, Korngi Karachi.

6. References

1. Karetzky S. Choosing an automated system, Library Journal. 1998; 123(11):42–44.
2. Reddy CS. Venkatarama .Comparative study of free /open source integrated library management system (FOSILMS) with reference to KOHA, NEWGENLIB and E-Granthalaya, E-Library Science Research Journal. 2012; 1(2):1–10.
3. Naveed e Sehar. Need and importance of automation and networking in public libraries of Karachi, Pakistan, Asian Journal of Multidisciplinary Studies. 2017; 5(7):1–8.
4. Pace A. 21st century library system, Journal of Library Administration. 2009; 49:641–46. https://doi.org/10.1080/01930820903238834.
5. Browser display statistics w3schools. Date accessed. 25.03.2019. https://www.w3schools.com/browsers/browsers_display.asp.
6. Sheikh, Gazala Shafiand Noman Islam. Towards a context -aware library Management system, International Journal of conceptions on Computing and Information Technolgy. 2015; 3(2):28–31. https://doi.org/10.1002/j.2637-496X.2015.tb00828.x.

7. Haider SJ. Library management scenario and management problems in Pakistani libraries, Library Administration and Management. 2007; 21(4):173–76.
8. De Smet E. ABCD: A new FOSS library automation solution based on ISIS, Information Development. 2009; 5(1):61–67. https://doi.org/10.2118/0909-0067-JPT.
9. Breeding M. The viability of open source ILS, Bulletin of the American Society for Information Science and Technology. 2009; 35(2):20–25. https://doi.org/10.1002/bult.2008.1720350207.
10. Breeding M. Opening up library automation software, Computers in Libraries. 2009; 29(2):25–27.
11. Tan H. C Language Programming. Tsinghua University Press, Beijing; 2005.
12. Touzi J, Benaben F, Pinguad H, Lorre JP. A model-driven approach for collaborative service-oriented architecture design, International Journal of Production Economics. 2009; 121(1):5–20.
13. Kinner L, Rigda C. The integrated library system: From daring to dinosaur? Journal of Library Administration. 2009; 49(4):401–17. https://doi.org/10.1080/01930820902832546.
14. Plugge Eelco, Membrey Peter, Hows David. Mongo DB Basics, USA:Apress; 2014. p. 144.
15. Laksono D. Testing Spatial Data Deliverance in SQL and NoSQL Database Using NodeJS. Fullstack Web App. In: 2018 4th International Conference on Science and Technology (ICST) IEEE. 2018 Aug 7; 1:1–5.
16. Zhang W, Zhao L. Creation and test of data access layer under Object/Relation mapping framework. In: 2012 IEEE International Conference on Information Science and Technology, IEEE; 2012 Mar 23. p. 270–73.